Brian Johnston
6/27/80

OVERVIEW OF DOS VERSION 2.4+

I. AREAS OF MAJOR CHANGE

1. Use of User's System RAM

The first thing a new DOS user is likely to notice is that when he powers on his computer with a Basic cartridge inserted, the Basic "Ready" prompt comes up much faster. The new "boot" is approximately six seconds as opposed to the original eleven seconds. This is because a significant number of DOS utilities are now stored in a separate file called "DUP.SYS" which is now loaded separately from DOS.SYS.

The FMS and portion of DOS which are loaded in the original boot we now call "Mini-DOS". Mini-DOS is stored on the system diskette under the name "DOS.SYS". The other file, "DUP.SYS", is loaded only when the user types "DOS". This has several positive ramifications.

A. A user has much less time to wait on the original system boot before the "Ready" prompt.
B. On minimal memory systems (16K) a significant number of bytes have been added to the available user area for running Basic programs, etc. This area was formerly used by the utilities now stored in the file DUP.SYS.
C. When the user types "DOS" the remaining utilities are now loaded and do overwrite a portion of the user's area. However the user can automatically protect his program and still use DOS by making use of a new "MEM.SAV" file. Mini-DOS saves the lower user area to be overwritten by DUP.SYS in MEM.SAVE whenever "DOS" is typed and then restores it whenever Basic is re-entered from DOS, if appropriate.

Although Mini-DOS does boot in faster than your old DOS and more RAM is available a price has been paid to achieve this. There are several negative ramifications.

A. A user must now wait for about five seconds for DUP.SYS to be loaded plus an additional twelve seconds if his user area is to be saved in MEM.SAV.
B. Having a MEM.SAV file will not protect user's RAM area beyond that needed for DUP.SYS. For this reason when using COPY FILE or DUPLICATE FILE options in DOS, the user must be careful to preserve the high user memory areaby not allowing DOS access to it. Otherwise the upper portion of user memory will be destroyed which will render the lower portion (saved in MEM.SAV) useless. Duplicate Disk cannot be used without destroying user memory and invalidating MEM.SAV.

```
**********************************************************
* WARNING:  Allowing DOS access to the user memory area is *
* always considered to have destroyed the upper user area. *
* Hence, MEM.SAV will not be reloaded on returning to the  *
* cartridge if user memory has been used by DOS.
**********************************************************
```

2. MEM.SAV File

A new menu item, "CREATE MEM.SAV", has replaced the old DEFINE DEVICE command. As has been previously discussed, once MEM.SAV exists on a diskette then lower user memory will be stored in MEM.SAV whenever DOS is called. The time required to accomplish this is 12 seconds (to save user memory) plus 5 seconds (to load in DUP.SYS). Thus if there is a MEM.SAV file the time to from boot into DOS is 6+5+12=23 seconds. Care must be taken not to allow DOS to use all of user memory when COPY FILE or DUPLICATE FILE commands are issued, however. Since DOS does not know if all of your Basic program has been saved, giving DOS permission to use all of user memory means that DOS must then invalidate any MEM.SAV file. And this means your program will not be reloaded on returning Basic. Thus will also be true if you execute the DUPLICATE DISK command which always uses the user memory area.

If there is no Basic program to protect (and no MEM.SAV) then Mini-DOS simply loads in DUP.SYS and the menu appears. Thus if there is no MEM.SAV file the total time to go from boot into DOS is 6+5=11 seconds which is approximately the same as before.

3. New Diskette Formats

A. Two DOS Versions.

All DOS Versions 2 have a new 3 sector boot which is required to handle the new dual density drive units. There will be two versions of the new DOS available however, one for single density drive only and one for both dual and single density drives. These versions will differ from each other only in the amount of buffer space they reserve.

The single density drive version of DOS must not be used with a dual density drive. This is because there are not enough buffers reserved for this version to work in all cases. Since dual drives require more buffer space it is not recommended that they be used with 16K systems though it will work. This is because the limited user RAM will be reduced by over 700 bytes if the dual density drive version is used.

B. Limitation On Duplicating Disks.

Another consequence of the new 3 sector boot is that the "J" command cannot be used to copy files from an old style diskette (whose boot routine is in one sector) onto a new style diskette (whose boot routine is in 3 sectors) and obtain a bootable new style diskette. This is due to the fact that the DUPLICATE DISK utility is a sector by sector copy routine and so the result is simply a duplicate old style diskette . Instead of duplicating from an old disk to a new disk (or vice-versa), one must use instead the COPY FILE or DUPLICATE FILE commands and transfer the files one at a time. Although this is slow, once the new disk is created it can then be duplicated using "J" very quickly and easily.

C. New Binary File Header.

A change has also been made to the header data at the beginning of every Binary File. Originally an 84(Hex), 09(Hex), the new header is an FF(Hex), FF(Hex). This allows a more flexible new load and go file implementation. With this change two Binary Files can be appended together using either COPY FILE or with SAVE BINARY FILE.

Suppose we wish to append File 2 to File 1. In this case we would use COPY FILE as follows:

```
SELECT ITEM
C |RETURN|
COPY-FROM,TO?
D2:FILE2,FILE1/A |RETURN|
SELECT ITEM
```

Suppose we wish to append a newly created 32 byte binary file in RAM to FILEA on our diskette. In this case we would use SAVE BINARY FILE as follows:

```
SELECT ITEM
K |RETURN|
SAVE-GIVE FILE,START,END[,INIT,RUN]
D2:FILEA/A,8000,801F |RETURN|
SELECT ITEM
```

In order to convert any pre-existing binary files over to the new header a utility "CNGLOAD.BAS" is available. This program is run as follows:

```
READY
RUN"D1:CNGLOAD.BAS |RETURN|
ENTER DRIVE NUMBER (IE 1,2,3,4)
?1 |RETURN|
ENTER FILE NAME
BFILE |RETURN|
CHANGE OF SIMPLE LOAD MODULE COMPLETE
READY
```

An error message will occur if you attempt to load an old style load file with any DOS Version 2.

4. New AUTORUN.SYS Files

"AUTORUN.SYS" is a reserved file name which allows the user to have any binary file of this name loaded and executed or simply loaded every time the system is booted. This might be anything from simple program to reset margins, to a program to enter DOS on boot (instead of the cartridge), to a complex user program or game.

EXAMPLE 1

An example of how to reset margins on boot using AUTORUN.SYS follows:

```
READY
POKE 82,3:POKE 83,38 |RETURN|
READY
DOS |RETURN|
SELECT ITEM
K |RETURN|
SAVE-GIVE FILE,START,END[,INIT,RUN]
AUTORUN.SYS,52,53 |RETURN|
SELECT ITEM
```

Remember that Basic commands use decimal numbers and DOS commands use hexidecimal numbers.

If you now turn your computer off and on again it will come up in Basic with new margins defined by your AUTORUN.SYS file.

EXAMPLE 2

An example of how to use AUTORUN.SYS to get your computer to always boot up into DOS follows next.

Any AUTORUN.SYS program which is not going to return to the DOS initialization routine that is normally executed following the execution of the AUTORUN file or which allows the use of SYSTEM RESET before returning to the initialization routine must finish the initialization before proceeding. This is done by modifying two Operating System storage locations, COLDST at address 244(Hex) and BOOT? at address 9(Hex). COLDST should be cleared to (00) and BOOT? is set to (01).

The program listed on the next page sets these two locations to the proper value and then jumps indirectly to the start DOS vector.

```
;Autorun Prog.
;
;Run DOS without going to cartridge.
;
COLDST=$244
BOOT=$09
DOSVEC=$0A
      *=$3000
;                              (HEX CODE)
DOSGO LDX #0                    A2 00
      STX COLDST                8E 44 02
      INX                       E8
      STX BOOT                  86 09
      JMP (DOSVEC)              6C 0A 00
      *=$2E2
      .WORD DOSGO
      .END
```

If you do not have any Assembler cartridge the equivalent file can be created using POKE in Basic and then saving the Binary File in DOS. The list of dec numbers to be entered is as follows:

(Dec Address)	(Dec Codes)
12288	162,00,
12290	142,68,02,
12293	232,
12294	124,09,
12296	108,10,00

When these codes have been entered in Basic using a series of POKES then the user enters DOS and saves the file using the following command.

```
SELECT ITEM
K
SAVE-GIVE FILE,START,END[,INIT,RUN]
AUTORUN.SYS,3000,300A,,3000
SELECT ITEM
```

If you turn off your computer and then turn it back on, you should now boot up directly into DOS. To enter Basic you simply press "SYSTEM RESET" or type "B |RETURN|".

II. CONSERVING MEMORY SPACE

1. Drive Buffer Allocation

The FMS reserves a 128 byte (or 256 byte) "drive buffer" for each disk drive you tell it is connected to your system. This occurs every time you boot up the system. The RAM location which controls how many drives FMS thinks are connected is Byte 1802(Dec). Basic can be used to Peek and Poke this location to put in the correct value for your system. You then enter DOS and write a new version of DOS, with the new disk information, out to your diskette. Now the new system will be automatically installed whenever you reboot your computer with the new DOS.

The meaning of the data in this cell is as follows:

Drive(s) Allocated	Drive Code
Drive 1 ----------	01(Dec)
Drive 2 ----------	02(Dec)
Drive 3 ----------	04(Dec)
Drive 4 ----------	08(Dec)
Also, Drive 1 + 2 ----------	03(Dec)
Drive 1, 2, 3, + 4-------	15(Dec)

If drives 1 and 2 are the new Dual-Dual Density Disk Drives, then FMS will automatically reserve a 256 byte buffer for each drive. If they are two single density disk drives then a 128 byte buffer will be reserved for each.

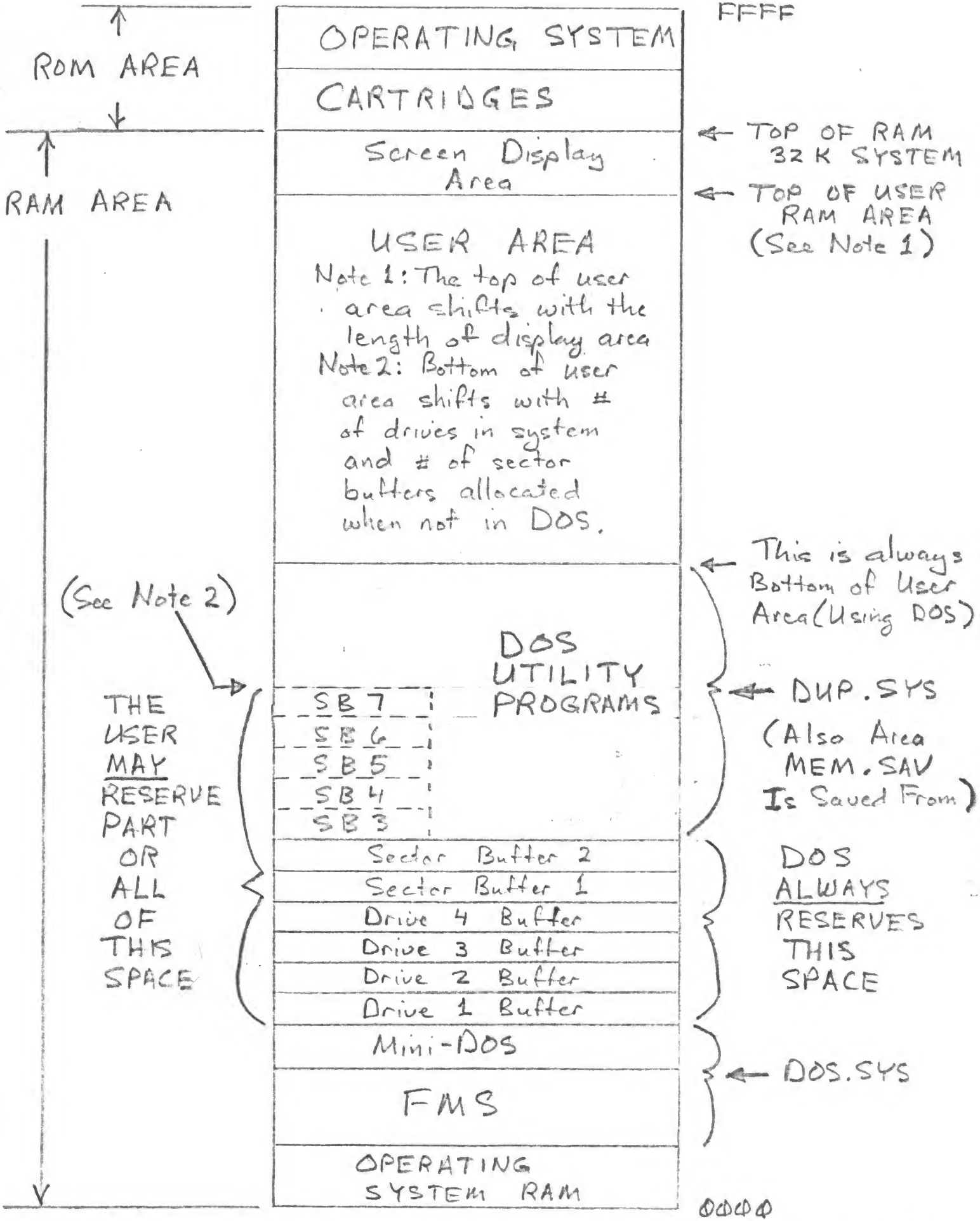
As before the system will only boot up from a drive with its switches set to the DRIVE 1 position.

2. Sector Buffer Allocation

FMS also reserves a 128 byte "sector buffer" (for single density) or a 256 byte sector buffer (for dual density) for every file that is to be open concurrently. RAM location 1801(Dec) contains a number which establishes these buffers when the computer is booted up. The number itself represents the number of 128 byte buffers which are reserved and can vary from 1 to 16(Dec). This means that a user could have up to eight files open (if all eight IOCB's were free for him to use). Since one IOCB is required for each open file one would never use more than eight buffers with single density drives. However, an open file on a dual density drive requires two adjacent 128 byte buffers (i.e. one 256 byte buffer) and thus eight open files on a dual density drive would require all sixteen 128 byte buffers.

RAM LOCATION 1801(Dec) CONTENTS	SINGLE DENSITY FILES OPEN	DUAL DENSITY FILES OPEN
1	1	-
2	2	1
3	3	-
4	4	2
5	5	-
6	6	3
7	7	-
8	8	4
9	-	-
10	-	5
11	-	-
12	-	6
13	-	-
14	-	7
15	-	-
16	-	8

DOS (Version 2.4+) itself requires that only two files must need to be open simultaneously. This means that DOS can safely utilize sector buffer areas that the user may have reserved for other purposes. Use of these sector buffer areas significantly enhances the speed of some DOS utilities such as COPY FILE, DUPLICATE FILE, and DUPLICATE DISK.



DOS VERSION 2 (Single Density Drives)

Brian Johnston
6/17/80

III. IMPROVEMENTS IN NEW FMS-DOS (Version 2.4+)

1. One can now use the Basic (or Assembler) RAM user program areas as a buffer to speed up processes like "COPY FILE" or "DUPLICATE FILE". This is especially valuable when long files are being worked on. If one needs to preserve the user program area then DOS can use a 250 byte buffer within its own RAM area for this purpose. You may find this considerably slower however, since much more overhead is required when copying long files with a 250 byte buffer.

2. A new command "CREATE MEM.SAV FILE" helps to make low memory systems such as a 16K Atari 400 or 800 much more powerful and allows one a smaller resident DOS. This new DOS gives you the ability to save any program in user RAM program areas in a file called MEM.SAV. The non-resident portion of DOS is "swapped" in following a DOS command by the user. Only the portion of RAM overlaid by the DUP.SYS utilities is stored in MEM.SAV to save time and disk space. To return to your original program environment from DOS, you simply type "B" RUN CARTRIDGE or press SYSTEM RESET. Remember, however, that you will wipe out a program stored in upper user program areas when copying or duplicating files if you give DOS permission to use <u>all</u> of user memory. Also giving DOS this permission will invalidate your MEM.SAV file. So be careful!

3. A new ROM for the single density disk drive has introduced a new format for diskettes which has contributed to speeding up many disk operations. A newly formatted diskette will work just as well on an old ROM disk drive as a new ROM disk drive. But, of course, an old ROM disk drive can only format in the old slower format. Any in-house drive labeled DISC C has this new ROM.

4. Improvements have been made which reduce system overhead for data block transfers.

5. Portions of SIO which could cause earlier versions of DOS to "hang up" when duplicating or copying a file using user memory have been replaced by new code included in FMS. This means that anyone booting up a Version 2.4+ DOS will not experience this problem with SIO.

6. The original disk handler (in ROM) had a problem which prevented the use of any write without read verify. A new disk handler has been written to work with both the new dual density disk drives and the old single density drives. With this new handler both, write with read verify and write without read verify, are available and have been included in the new FMS.

7. The object file header for binary load files has been changed from 84H,09H to FFH,FFH. This modification is consistent with the new Assembler cartridge and allows for a flexible new type of "load and go" files to be discussed next. A utility is available to enable you to convert any binary file created under the original DOS to this new format.

8. The "SAVE BINARY FILE" offers new power and flexibility. Optional parameters are a new initial and final run address. As before binary files can be appended to one another using the "/A" option. If no initial or final run address is specified then each binary file will simply be a "load type file". Similarly a series of such files appended together will also be a new load type file since each of its components is a load type file.

   If an initial run address is specified, then, as soon as the file is loaded, it will be executed. This is a "load and go" type file. If more than one "load and go" type is appended together then each file will load and execute before the next file gets loaded. "Load files" and "load and go files" can be mixed together.

   If a final run address is specified, then once again the file will be executed. However, this time the run address will not be jumped to until everything in the file is completely loaded into memory. For example suppose we have a compound file where a load file "B" is appended to another load file "A" which has a final run address specified. In this example "A" would load first but not execute, then "B" loads and finally "A" executes.

9. The new DOS also has a special response to any file called "AUTORUN.SYS". Whenever the computer is booted up, if a file by this name exists on the disk, the AUTORUN.SYS file will be loaded and/or executed before returning control.

10. The "LOAD BINARY FILE" command has a new option "/N" which allows you to load a "load and go" type file without having it execute.

11. There is no longer an irritating delay when a file is opened to append as the operating system links through each sector to find the end of the file. The new procedure does not link new data into an old file now until the modified file is closed. This means that in many instances the delay is much less noticable though still present.

12. The "COPY FILE" command has an "/A" option which allows any two files to be appended to each other. This may or may not cause problems depending on whether the two files are complimentary. For example, two Basic files with similar line numbers could interfere with each other when merged together.

    Note: It does not make sense to append "Tokenized" Basic files together as each has its own symbol table etc. and only the first file will be written. However, "List" type Basic files can be merged and "Binary" files created by DOS (or by the Assembler Cartridge) can also be merged.

13. Duplicate Disk now uses full user memory for duplicating diskettes from one drive to another as well as on the same drive. This has significantly speeded up the duplication process for nearly full diskettes. The more user RAM available, the greater the speed up will be.

14. Formatting a diskette now gives you an error message when the format fails. Suggested procedure is to retry the format and if it fails three or more times, the diskette should be discarded as it has bad sectors on it.

15. Margins are now reset on entering DOS.

16. Full length filenames are now accepted in all instances.

17. Use of wild cards in "COPY FILE" and "DUPLICATE FILE" commands has been prohibited although this feature may be installed in a future release.

18. This version of DOS will allow you to use both the old single density disk drives and the new dual density disk drives when they become available.

    Note: It is not possible to use the "DUPLICATE DISK" command to copy from an old diskette created with a Version 1 DOS (9/24/79) to a Version 2 DOS. This is because your Version 2 DOS has a three sector boot instead of Version 1's one sector boot. This new boot loads a new disk handler routine which allows Version 2 DOS to handle the new Dual Density Disk Drives.

19. Occasional random errors in duplicating the first eight sectors of a diskette have been eliminated. One does not need to rewrite DOS.SYS after duplicating a diskette now to be sure of having a bootable version of DOS. This problem arose from DOS usually being located in the first eight sectors.